Homework 11

Charles Swarts
Swarts2

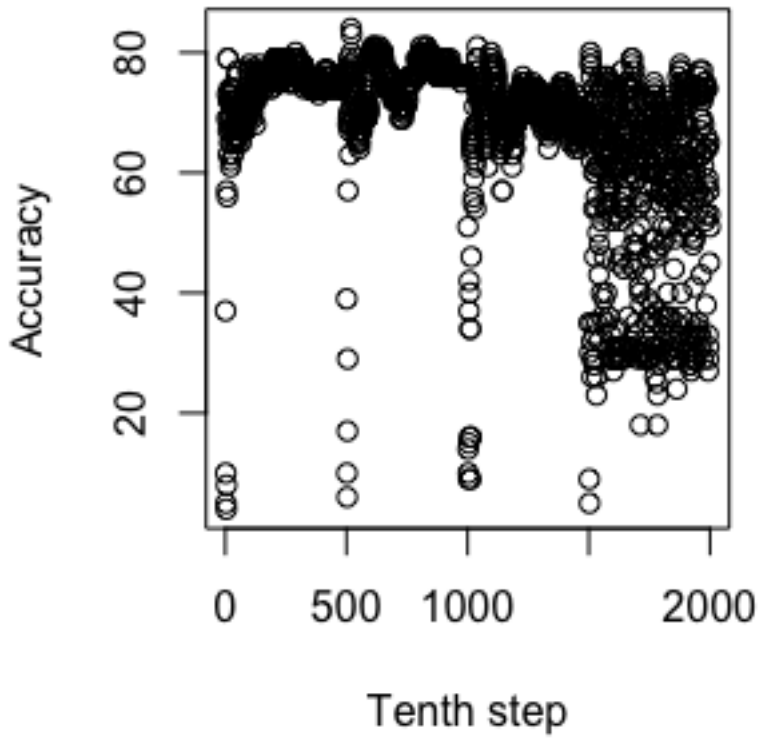
April 2016

1 12.1

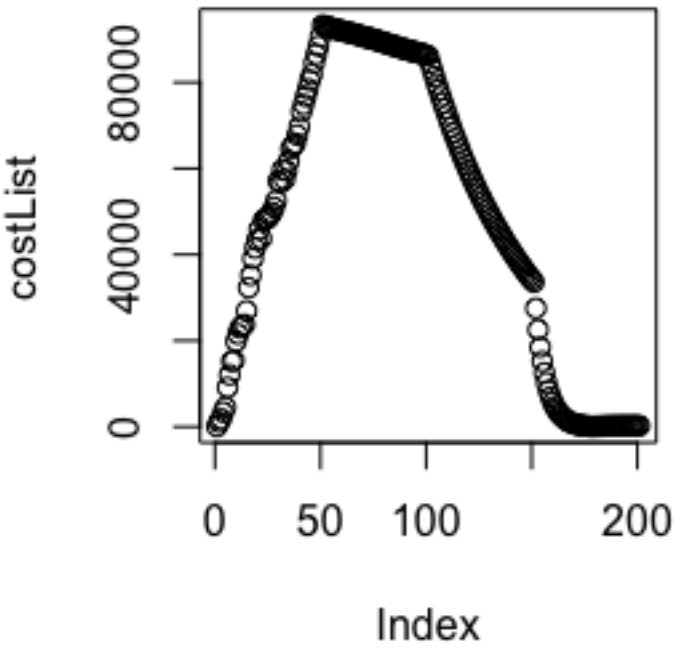
1.1 question

The UC Irvine machine learning data repository hosts a collection of data on breast cancer diagnostics, donated by Olvi Mangasarian, Nick Street, and William H. Wolberg. You can find this data at http://archive.ics.uci.edu/ ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic). For each record, there is an id number, 10 continuous variables, and a class (benign or malignant). There are 569 examples. Separate this dataset randomly into 100 validation, 100 test, and 369 training examples.
Write a program to train a support vector machine on this data using stochastic gradient descent. You should not use a package to train the classifier (you don't really need one), but your own code. You should ignore the id number, and use the continuous variables as a feature vector. You should search for an appropriate value of the regularization constant, trying at least the values =[1e3,1e2,1e1,1]. Use the validation set for this search
You should use at least 50 epochs of at least 100 steps each. In each epoch, you should separate out 50 training examples at random for evaluation. You should compute the accuracy of the current classifier on the set held out for the epoch every 10 steps. You should produce:
(a) A plot of the accuracy every 10 steps, for each value of the regularization constant.
(b) Your estimate of the best value of the regularization constant, together with a brief description of why you believe that is a good value.
(c) Your estimate of the accuracy of the best classifier on held out data

1.2 answer



(a)The distinct sections of the graph are by regularization value of increasing order.



(b)This was my first attempt at getting the machine to learn. The four sections represent each of the regularization constants. Based on this graph, the last section, representing = 1 is the best out of the four by far for getting the cost low. However, a low cost is actually poorly correlated with accuracy as shown above. Therefore I am going to say the .01 regularization constant is the best.
(c)When it's accurate on the validation set, its about just as accurate on the test set. So when I got it to be 85% accurate in the validation set and it came out to be about 82% accurate on the test case. So with the regularization constant set to .01, it's about 82% accurate.

1.3 raw code

```
    #Read in the data.

BCW <- read.csv("/Users/CharlesBSwarts/Desktop/BCW.csv",sep=",",header=FALSE)
head(BCW)

#Standardize the data

for(i in 3:ncol(BCW))
{
  BCW[,i] <- (BCW[,i]-mean(BCW[,i]))/sd(BCW[,i])
}

#Randomly select the test, validation,
```

```
#and training sets.

space<- 1:nrow(BCW)
training <- sample(space,369,replace=FALSE)
space <- setdiff(space,training)
validation <- sample(space,100,replace=FALSE)
testing <- setdiff(space,validation)


#Set and initialize important variables
#including the a vector and the b variable

costList <- c()
accuracyList <- c();
accuracy <- 0
lambdas <- c(.001,.01,.1,1)
lambda <- lambdas[2]
epochs <- 50
stepspepoch <- 100
a <- runif(30,-.1,.1)
b=.1
for(i in 1:epochs)
{
  #At the beginning of each epic
  #I set the step size, and I
  #pick the data points to learn
  #from.
  stepSize <- 1/(.01*i+50)
  yvalues <- sample(training,100,replace=TRUE)
  for(i in 1:stepspepoch)
  {
    #I figure out whether it's
    #a null or not
    y <- BCW[yvalues[i],2]
    if(BCW[i,2]=="M"){y <- 1}
    if(BCW[i,2]=="B"){y <- -1}

    #Then I use these formulas from the book.
    if(y*((a)%*%as.numeric(BCW[yvalues[i],3:32])+b)>=1)
    {
      a <- a-stepSize*lambda*a
    }
    if(y*((a)%*%as.numeric(BCW[yvalues[i],3:32])+b)<1)
    {
      a <- a-stepSize*(lambda*a-y*as.numeric(BCW[training[i],3:32]))
      b <- b+stepSize*y
    }

    #At every tenth step, I calculate the
    #accuracy so far.
    if(i%%10==0)
    {
```

```
      for(j in 1:100)
      {
        if(BCW[validation[j],2]=="M"){y <- 1}
        if(BCW[validation[j],2]=="B"){y <- -1}
        if(y*(a%*%as.numeric(BCW[(validation[j]),3:32])+b)>0)
        {accuracy <- accuracy+1}
      }
      accuracyList <- c(accuracyList,accuracy)
      accuracy <- 0
    }

  }
  #At the end of each epoch, I
  #calculate the total cost that
  #I am getting with the current
  #classifier.

  cost <- 0
  for(k in 1:100)
  {
    if(BCW[validation[k],2]=="M"){y <- 1}
    if(BCW[validation[k],2]=="B"){y <- -1}
    #print(1-y*sum((t(a)*BCW[(validation[i]),3:32])))
    if(y*(a%*%as.numeric(BCW[(validation[k]),3:32])+b)>=1)
    {cost <- cost}
    if(y*(a%*%as.numeric(BCW[(validation[k]),3:32])+b)<1)
    {cost <- cost+ (1-y*(a%*%as.numeric(BCW[(validation[k]),3:32])))}
  }
  costList <- c(costList,cost)

}

#A nice plot of accuracy.

accuracyList
plot(accuracyList,ylab="Accuracy",xlab="Tenth step")
plot(costList)


#Then I test the cost of the model
#based on the test data.

costListTest <- c()
costTest <- 0
for(k in 1:100)
{
  if(BCW[testing[k],2]=="M"){y <- 1}
  if(BCW[testing[k],2]=="B"){y <- -1}
  if(y*(a%*%as.numeric(BCW[(testing[k]),3:32])+b)>=1)
  {costTest <- costTest}
  if(y*(a%*%as.numeric(BCW[(testing[k]),3:32])+b)<1)
  {costTest <- costTest+ (1-y*(a%*%as.numeric(BCW[(testing[k]),3:32])))}
```

```
}
costListTest <- c(costList,costTest)
costListTest

# Finally I test how accurate the
# classifier is.

accuracyTest <- 0
for(j in 1:100)
{
  if(BCW[testing[j],2]=="M"){y <- 1}
  if(BCW[testing[j],2]=="B"){y <- -1}
  if(y*(a%*%as.numeric(BCW[(testing[j]),3:32])+b)>0)
  {accuracyTest <- accuracyTest+1}
}
accuracyTest
```